PEP Scala (2)

Email:	christian.urban at kcl.ac.uk
Office:	N7.07 (North Wing, Bush House)
Slides & Code:	KEATS

Scala Office Hours:	Thursdays 11 – 13

Mea Culpa

CW6, Part 3 (deadline 21 December)

```
val blchip_portfolio =
 List("GOOG", "AAPL", "MSFT", "IBM", "FB",
                              "YHOO", "AMZN", "BIDU")

val rstate_portfolio =
List("PLD", "PSA", "AMT", "AIV", "AVB",
                              "BXP", "CBG", "CCI",
     "DLR", "EQIX", "EQR", "ESS", "EXR",
                              "FRT", "GGP", "HCP")
```

Mea Culpa

CW6, Part 3 (deadline 21 December)

```
val blchip_portfolio =
 List("GOOG", "AAPL", "MSFT", "IBM", "FB",
                              "YHOO", "AMZN", "BIDU")

val rstate_portfolio =
List("PLD", "PSA", "AMT", "AIV", "AVB",
                              "BXP", "CBG", "CCI",
     "DLR", "EQIX", "EQR", "ESS", "EXR",
                              "FRT", "GGP", "HCP")
```

The results in the CW are calculated with YHOO and CBG deleted.

Mea Culpa 2

Avoid at all costs, even in comments

- `var`
- `return`
- `.par`
- `ListBuffer`
- `mutable`

Mea Culpa 2

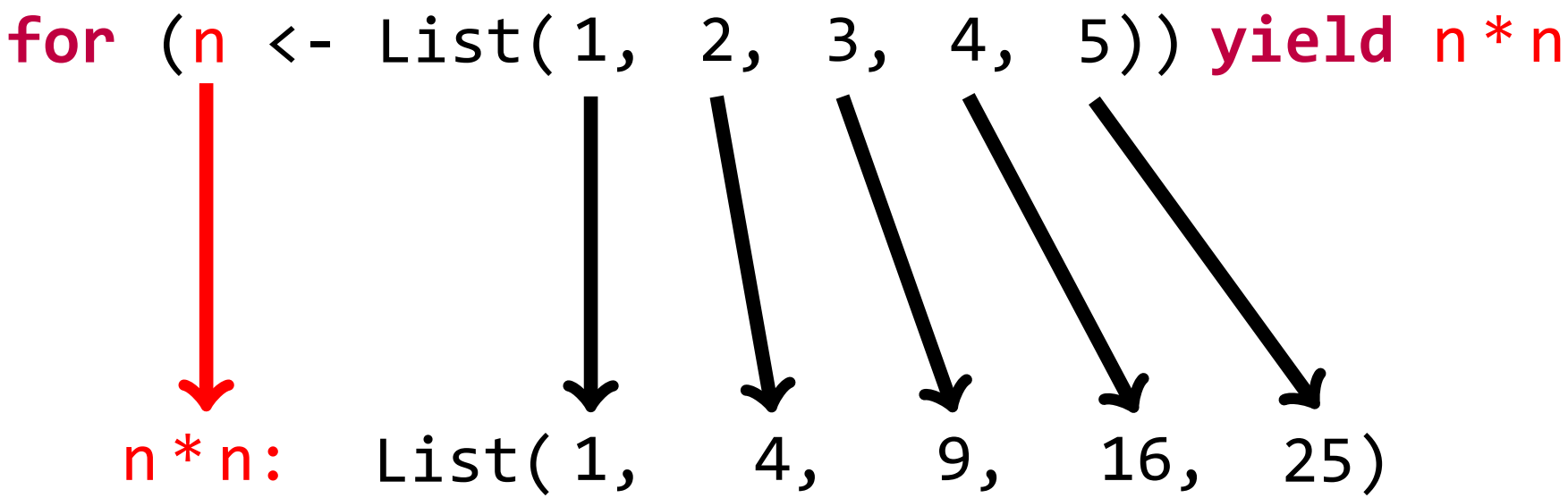
Avoid at all costs, even in comments

- `var` $\Rightarrow$ `Var`
- `return` $\Rightarrow$ `Return`
- `.par`
- `ListBuffer`
- `mutable`

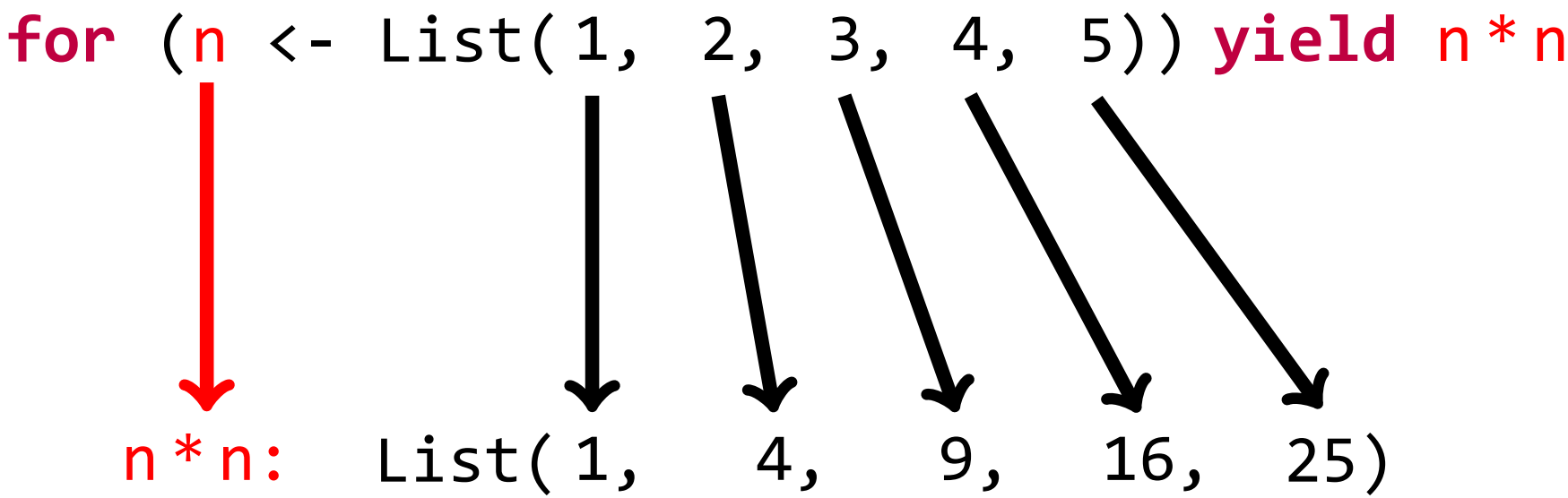
For-Comprehensions Again

```
for (n <- List(1, 2, 3, 4, 5)) yield n*n
```

For-Comprehensions Again



For-Comprehensions Again



This is for when the for-comprehension **yields / produces** a result.

For-Comprehensions Again

```
for (n <- List(1, 2, 3, 4, 5)) yield n*n
```

vs

```
for (n <- List(1, 2, 3, 4, 5)) println(n)
```

The second version is in case the for **does not** produce any result.

Why Scala?

- **You can avoid `null`:**

> "I call it my billion-dollar mistake. It was the invention of the null reference in 1965. At that time, I was designing the first comprehensive type system for references in an object oriented language (ALGOL W). My goal was to ensure that all use of references should be absolutely safe, with checking performed automatically by the compiler. But I couldn't resist the temptation to put in a null reference, simply because it was so easy to implement. This has led to innumerable errors, vulnerabilities, and system crashes, which have probably caused a billion dollars of pain and damage in the last forty years." Sir Tony (Hoare)

Types

- Base types

  `Int`, `Long`, `BigInt`, `Float`, `Double`
  `String`, `Char`
  `Boolean`

- Compound types

`List[Int]`	lists of Int's
`Set[Double]`	sets of Double's
`(Int, String)`	Int-String pair
`List[(BigInt, String)]`	lists of BigInt-String pairs
`List[List[Int]]`	list of lists of Int's

Questions?